Dipartimento di Informatica

Laurea Specialistica in Informatica

Progettazione di Interfacce e Valutazione dell'Usabilità

Applicazione Web multi-dispositivo "Rate Your Life"

Autore: Luca Cito
Docente: Fabio Paternò
A.A.: 2014/2015
Data: 18 febbraio 2015

Indice

1 Introduzione **2**

2 Requisiti **3**
2.1 Obiettivi 3
2.2 Applicazioni simili 3
2.3 Scenari d'uso 3
2.4 Specificazione dei requisiti 4
2.5 Principi durante la progettazione 5

3 Implementazione **6**
3.1 Tecnologie e framework 6
3.1.1 Librerie utilizzate 7
3.2 Breakpoints e layout 9
3.3 Overview + Detail 11
3.3.1 Implementazione 11
3.4 Fisheye 13
3.4.1 Implementazione 14
3.5 Gestione dei tag 15

4 Valutazione dell'usabilità **16**
4.1 Metodo di valutazione 16
4.2 Risultati 16

5 Conclusione **19**

1 Introduzione

Gli ultimi anni, fin dall'introduzione del primo iPhone nel 2007, sono stati all'insegna della diffusione di dispositivi in grado di accedere all'internet 24 ore su 24. Questo ha comportato molti cambiamenti nell'ambito della progettazione di interfacce: come sviluppare una applicazione, come per esempio un sito web, che deve essere *usabile* in contesti completamente diversi? Le possibilità di layout, le modalità di interazione e molti altri aspetti devono tener conto di questa dualità, che a tratti appare quasi una dicotomia.

Durante il corso *Progettazione di interfacce e valutazione dell'usabilità* sono stati presentati diversi concetti per risolvere queste domande; la seguente relazione tenta di mostrarne la realizzazione nell'ambito di un'applicazione web multi-dispositivo. *Rate Your Life* nasce dall'idea di un diario online che permette di raccogliere statistiche sulla propria vita e di tracciarne lo sviluppo con diverse metriche. RYL sarà il prototipo su cui verranno applicati i concetti di progettazione di interfacce.

2 Requisiti

In questo capitolo vengono presentati i requisiti dell'applicazione. Dopo un elenco degli obbiettivi principali e una ricerca di applicazioni simili, vengono descritti gli scenari d'uso più importanti. Questi permettono di definire una specificazione dei requisiti semi-formale su cui basarsi durante l'implementazione.

2.1 Obiettivi

L'obbiettivo di RYL è permettere all'utente di dare un voto alla proprio giornata e di rivedere con facilità i voti dati in passato. Il punto focale del progetto rimane però l'aspetto multi-dispositivo dell'applicazione, che deve permettere all'utente di votare sia da un *desktop* che da un dispositivo *mobile*, in modo di poter votare e, se necessario, cambiare il voto in qualunque momento durante il corso della giornata.

2.2 Applicazioni simili

Una ricerca di applicazioni simili ha dato pochi frutti. Il concetto di monitoraggio è molto diffuso tra i cosiddetti dispositivi *wearables*, dove però ci si concentra più su funzioni vitali e metriche "fisiche" dell'utente, che su come lui veda la propria giornata. In questo contesto trovano più riscontro blogs e social networks, i quali permettono studi interessanti[1] al confine tra la psicologia comportamentale e il data mining. Questo tipo di applicazioni non presentano però, a differenza del prototipo di RYL, metriche numeriche che l'utente può scegliere consapevolmente.

2.3 Scenari d'uso

I due scenari principali sono la visualizzazione di un giorno e la possibilità di assegnargli un voto. In aggiunta, è possibile gestire i tag in una maschera appropriata. In questo stadio iniziale si considera l'utente del sito come unico tipo di attore.

- **Votare:** l'utente accede al sito e si ritrova sulla sua homepage. Dopo aver selezionato il giorno desiderato, assegna un voto e scrive un breve commento per motivare la valutazione. Dopodiché contrassegna il giorno con dei tag scelti dalla lista di tag disponibili. Alcuni li contrassegna come positivi, altri invece li

[1] `http://www.pnas.org/content/111/24/8788.full`

mette nella categoria di cose che hanno avuto una influenza negativa sulla sua giornata.

- **Visualizzare un giorno:** l'utente accede al sito e si ritrova sulla sua homepage. Dopo aver selezionato il giorno desiderato, gli viene mostrato il voto che aveva assegnato precedentemente, oltre alla lista di tag nelle categorie appropriate.
- **Gestire i tag:** l'utente accede al sito e si ritrova sulla sua homepage. Siccome la lista di tag disponibili non gli sembra soddisfacente, decide di cancellarne alcuni, modificare il nome di altri e infine di aggiungerne un paio, in modo da poter descrivere meglio le sue giornate.

E' da accennare che purtroppo nel prototipo il tempo non è bastato per implementare una funzione di login, la quale manca quindi dagli scenari d'uso. Ciononostante, si tratta di una modifica al modello di dati dell'applicazione relativamente semplice e la sua mancanza non sminuisce la rilevanza di questo caso di studio.

2.4 Specificazione dei requisiti

Dagli scenari d'uso appena elencati è possibile ricavare una lista di requisiti dell'applicazione. Questi si dividono in requisiti funzionali (i compiti veri e propri di RYL) e non funzionali (le proprietà che il sistema deve mantenere durante l'esecuzione). Siccome il progetto è stato sviluppato nel contesto della progettazione di interfacce, le proprietà del sistema hanno un valore maggiore delle funzionalità stesse.

1. Requisiti funzionali:
   a) L'utente può selezionare un giorno. Se per il giorno selezionato è già stato assegnato un voto, un commento o dei tag, questi devono essere visualizzati.
   b) L'utente può assegnare un voto da 1 a 10 al giorno selezionato. Ogni voto è *unico* per il giorno a cui è assegnato.
   c) L'utente può assegnare un commento al giorno selezionato. Il commento è opzionale e non deve eccedere una lunghezza di 140 caratteri.
   d) L'utente può assegnare dei tag al giorno selezionato. Ogni giorno può avere molteplici tag e ogni tag può essere assegnato arbitrariamente a molteplici giorni. Nel contesto dell'assegnazione di un tag ad un giorno, l'utente può specificare se l'assegnamento è di natura positiva o negativa.
   e) Tutti i voti e tutte le assegnazioni sono editabili.
   f) I tag possono essere creati, visualizzati, modificati e rimossi (cosiddette funzionalità CRUD).

2. Requisiti non funzionali:
    a) Usabilità: per ovvi motivi il requisito centrale dell'applicazione. I temi di grafiche responsive, affordance e le varie tecniche presentate durante il corso devono trovare uso in RYL.
    b) Capacità multi-dispositivo: l'applicazione deve essere usabile sia da desktop che da mobile. Tutte le visualizzazioni e funzionalità devono tenere conto delle varie dimensioni di schermo e dei tipi di interazioni possibili.

Fig. 2.1 mostra il modello dati ricavato dai requisiti.

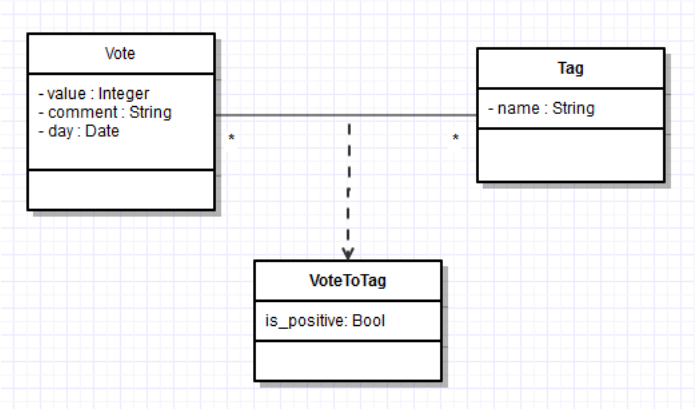


Figura 2.1: Modello dei dati di RYL

2.5 Principi durante la progettazione

Soprattutto per realizzare il requisito non funzionale dell'usabilità, durante la progettazione dell'applicazione sono stati seguiti alcuni principi fondamentali:

- Semplicità: RYL minimizzerà il testo presente e si concentrerà su un layout intuitivo per l'utente. Uno dei concetti principali in questo campo è quello dell'*affordance*: la proprietà percepibile degli oggetti che indica come potrebbero essere usati. Tramite l'uso di metafore, l'utente deve essere portato a poter usare l'applicazione senza spiegazioni.

- Dinamicità: l'uso dell'applicazione deve essere fluido e senza interruzioni. Perciò verrà evitato, dove possibile, l'uso di tecniche che obbligano il browser a ricaricare la pagina, come per esempio le forms.

- Estetica moderna: il sito verrà progettato orientandosi all'estetica dei siti attuali, partendo da un uso possibilmente uniforme dei colori, fino al layout "a rete" tipico delle applicazioni moderne.

3 Implementazione

Dopo aver elencato i requisiti a cui deve adempiere RYL, questo capitolo descriverà i dettagli dell'implementazione, partendo dal framework scelto fino ad arrivare alla spiegazione delle varie tecniche di usabilità applicate.

3.1 Tecnologie e framework

Per lo sviluppo del progetto è stato scelto Django, un web framework basato su Python. Come ambiente di sviluppo è stato usato JetBrains PyCharm.

Python

Python[1] è un linguaggio di programmazione ad alto livello. Viene usato principalmente in campo accademico per computazioni numeriche e per scripting di vario tipo (avendo ereditato il posto di Perl). E' un linguaggio dinamico e interpretato che permette di descrivere procedure complesse in maniera concisa (a differenza di altri linguaggi popolari come Java) grazie a vari tipi di astrazioni disponibili, tra cui molti concetti della programmazione funzionale. Python offre inoltre un sistema di gestione centrale di librerie, *pip*[2], che facilita molto lo sviluppo. Negli ultimi anni, Python ha riscontrato molto successo nel campo dei framework per applicazioni web, tra cui Flask, Pyramid e Django.

Django

Django[3] è un framework che si basa sul modello MVC (model-view-controller) e rispetta come concetto fondamentale il *convention over configuration*: l'uso di diverse convenzioni per permettere allo sviluppatore di "fidarsi" del framework per gli aspetti generici più comuni e potersi concentrare su parti specifiche dell'applicazione. Alcune di queste funzionalità comprendono un'astrazione per interagire con il database e view generiche per funzionalità CRUD. Grazie a questo e alle librerie disponibili, Django permette di sviluppare prototipi molto velocemente. Una lista dei siti che usano Django comprende *Instagram* e i siti della *NASA* e *The Guardian*.

[1] `https://www.python.org/`
[2] `https://pypi.python.org/pypi/pip`
[3] `https://www.djangoproject.com/`

3.1.1 Librerie utilizzate

Durante il progetto sono state usate varie librerie per facilitare l'implementazione di alcune funzionalità, soprattutto nel front-end.

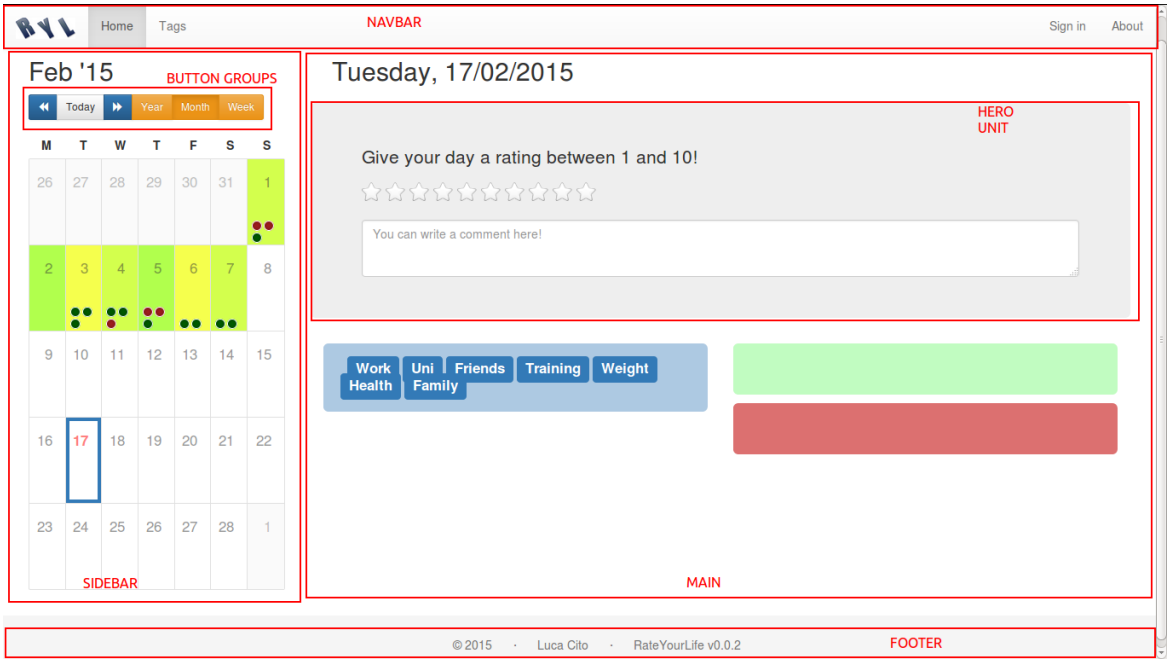
jQuery

jQuery[4] è la libreria per client-side Javascript più diffusa. Offre una varietà di funzioni che facilitano l'uso di Javascript nel browser, in modo da poter rendere l'applicazione più dinamica. Oltretutto permette di separare i tag HTML dal puro codice Javascript, dividendo la parte statica del sito dal resto. RYL usa jQuery anche per le varie comunicazioni con il server tramite Ajax.

Bootstrap

Bootstrap[5] è una libreria per il front-end, concentrata soprattutto su CSS. Unisce varie *best practices* dello sviluppo di applicazioni web, offrendo per esempio varie classi CSS che facilitano il layout di siti responsive. Fig. 3.1 illustra alcune componenti di Bootstrap e il loro uso nella homepage dell'applicazione.
Il layout di Bootstrap segue quello di una rete. Lo schermo è implicitamente suddiviso in 12 colonne, e ad ogni elemento dell'applicazione possono essere assegnate delle classi

Figura 3.1: Elementi di Bootstrap nella Home di RYL



[4] http://jquery.com/
[5] http://getbootstrap.com/

che definiscono quale percentuale dello schermo può coprire. Per esempio, il layout della homepage (v. 3.4) è definito in `home.html` in questo modo:

```
<div id="calDiv" class="col-xs-12 col-sm-6 col-md-3 cal">
   {% include "RateYourLife/calendar.html" %}
</div>
<div id="dayDiv" class="col-xs-12 col-sm-6 col-md-9">
   {% include "RateYourLife/day.html" %}
</div>
```

Questo vuol dire che per i dispositivi desktop (`col-md-*`) il calendario occupa $^1/_3$ dello schermo e la schermata del giorno i restanti $^2/_3$. Le altre classi definiscono il comportamento della schermata per risoluzioni più basse, e vengono spiegate in dettaglio in 3.2.

Bootstrap Calendar

I giorni sono un elemento fondamentale del modello dei dati dell'applicazione, il che rende necessario un metodo per visualizzarli in maniera organizzata. Il sistema più ovvio è il calendario. La libreria *Bootstrap Calendar*[6] è un implementazione che segue i principi di layout di Bootstrap, rimanendo quindi consistente con il resto del sito. La libreria è stata fortemente modificata per essere accomodata alle necessità di RYL, in accordo con la licenza MIT con cui viene distribuita.

jQuery Raty

Seguendo il principio della affordance, nel campo di voti e valutazioni una delle metafore usate più spesso è quella delle stelle. La maggior parte dei siti, da Amazon a Goodreads, che contengono la possibilità di assegnare un voto, usano stelle per indicare questa funzione all'utente. RYL usa il jQuery-plugin *Raty*[7] per visualizzare delle stelle nella schermata di voto.

breakpoints.js

Il requisito 2b) specifica la necessità di un design che si adatta a diverse larghezze dello schermo. Data l'interazione tra Javascript, CSS e HTML presente nell'applicazione, non è bastato l'uso di media queries. Si è fatto perciò ricorso alla libreria *breakpoints.js*[8]: essa riceve una serie di breakpoints e genera automaticamente degli eventi quando si passa da una grandezza all'altra. Questi eventi possono essere catturati tramite jQuery per cambiare il comportamento dell'applicazione. Per una lista dei breakpoints utilizzati e dei loro effetti, v. 3.2.

[6]Disponibile su Github: `https://github.com/Serhioromano/bootstrap-calendar`
[7]Disponibile su Github: `https://github.com/wbotelhos/raty`
[8]Disponibile su Github: `https://github.com/xoxco/breakpoints`

3.2 Breakpoints e layout

I breakpoints usati sono a 640, 768 e 992 pixel:

- 640px: la risoluzione per dispositivi mobili. Qui l'unico elemento presente è il calendario, mentre la barra di navigazione in alto viene collassata e il piè pagina viene nascosto. Fig. 3.2 mostra la homepage a questa risoluzione. Per i dispositivi mobili, il calendario è dotato di una funzione *fisheye*, che viene spiegata dettagliatamente in 3.4.
- 768px: la risoluzione intermedia. Qui la homepage è progettata secondo il layout *overview+detail*, che viene analizzato in 3.3. Per evitare che il calendario diventi troppo piccolo, sia la panoramica che il dettaglio sono definiti con `col-sm-6`, in modo da occupare ciascuno metà dello schermo. Fig. 3.3 mostra il layout a questa risoluzione.
- 992px: la risoluzione per desktop. Come visibile in Fig. 3.4, il dettaglio ora occupa $^2/_3$ della schermata. La barra di navigazione in alto non è più collassata ed il piè pagina è visibile. Inoltre, le stelle usate nel plugin Raty sono più grandi.

Dispositivi supportati

In generale è sconsigliabile basarsi su dispositivi specifici durante la progettazione di un'applicazione multi-dispositivo: non si può sapere quale tecnologie possono diventare obsolete e quali trend possono inaspettatamente avere successo. Per questo l'approccio più sensato è quello del *mobile first*: partire dalla progettazione per lo schermo più piccolo disponibile e aggiungere gli elementi passando a risoluzioni più alte. Di conseguenza i dispositivi supportati sono tutti quelli superiori ai 320 pixel. Durante i test sono stati usati un portatile con una risoluzione 1366x769 pixel e uno smartphone con una risoluzione di 640x1280 pixel.

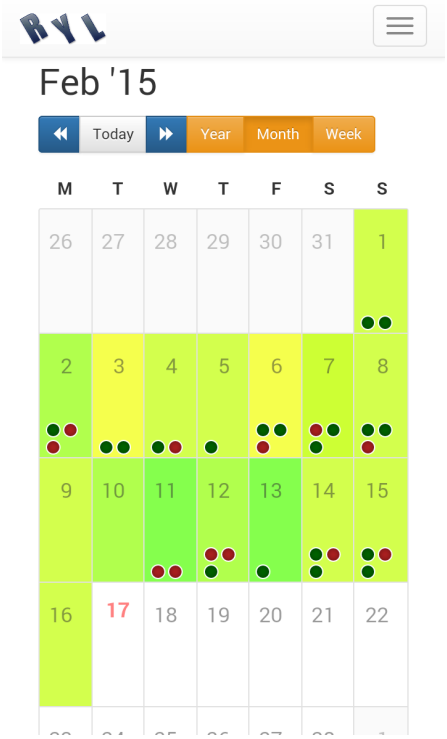


Figura 3.2: Layout a 640 pixel

Browser supportati

I browser supportati sono Mozilla Firefox 35 e Google Chrome 40. Non trattandosi di un applicazione da ufficio e volendo l'autore evitare traumi avvenuti in passato, Internet Explorer non è supportato.

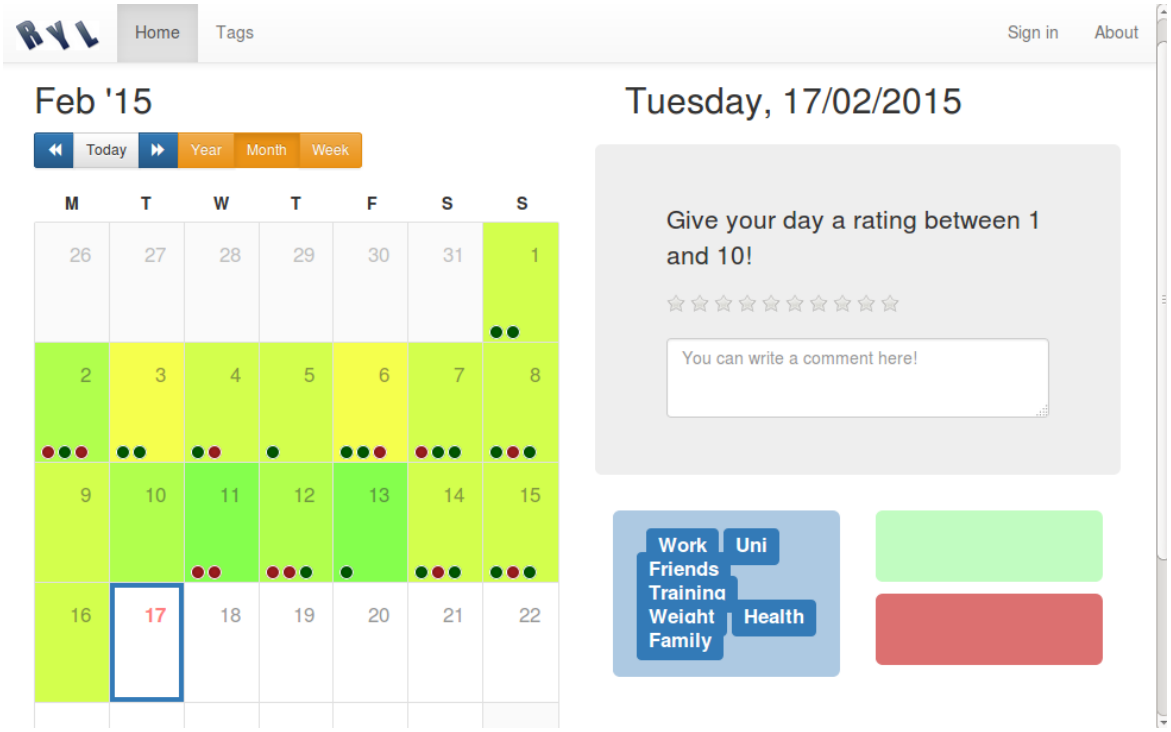


Figura 3.3: Layout a 768 pixel

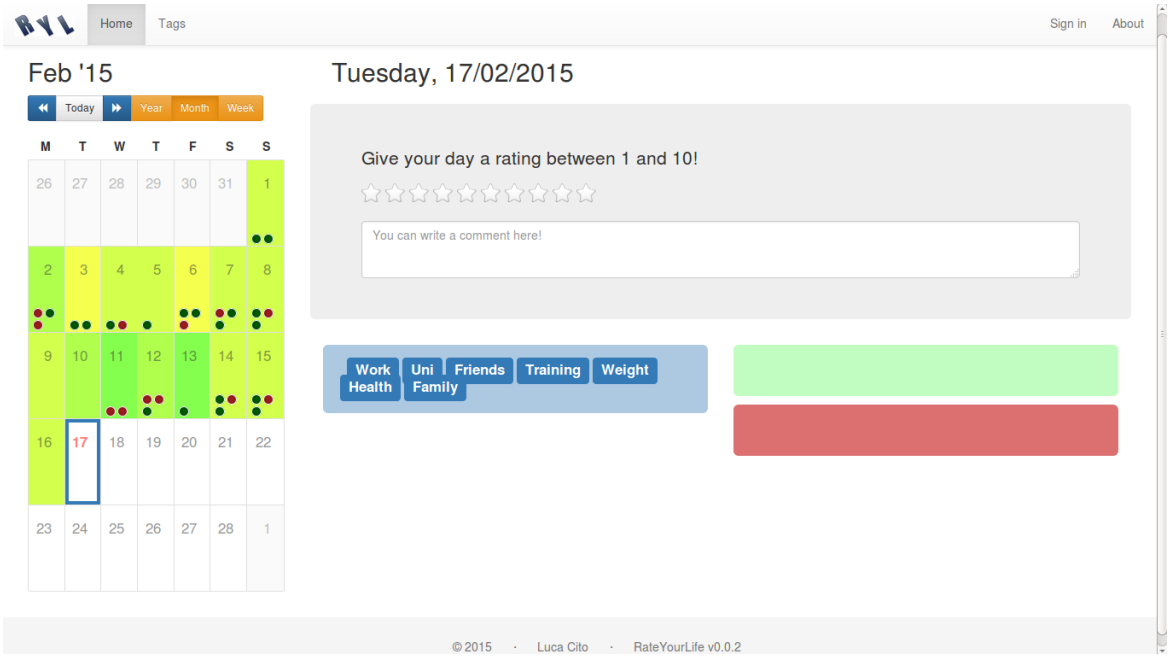


Figura 3.4: Layout a partire da 992 pixel

3.3 Overview + Detail

Il design *overview+detail*[9] è concepito per offrire due visualizzazioni simultanee di spazio informativo. Il dettaglio (detail) viene selezionato e evidenziato sulla panoramica (overview). Un esempio di overview+detail è Google Street View, in cui la mappa e la foto della strada rappresentano simultaneamente lo stesso posto.
Overview+detail offre una separazione spaziale e permette così all'utente di visualizzare contemporaneamente un elemento da due prospettive.

3.3.1 Implementazione

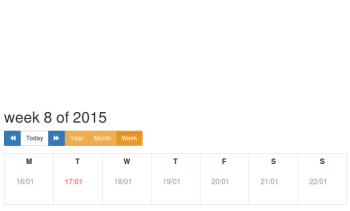
RYL implementa il design overview+detail separando, come mostrato in 3.2, la homepage in due parti diverse: il calendario a sinistra e la maschera del voto a destra; in questo il calendario funge da overview, mentre la visualizzazione del giorno ha la funzione di detail.

Calendario

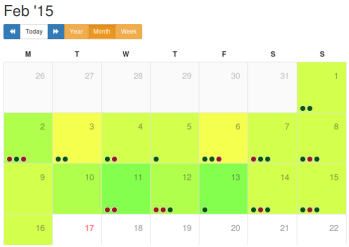
Nel calendario è possibile cambiare la visualizzazione tra quella in anni, in mesi e in settimane (v. Fig. 3.5). Il titolo si adatta alla visualizzazione scelta ed è possibile navigare tramite i bottoni in alto. I giorni del calendario sono divisi in varie categorie:

- Giorni non appartenenti al mese selezionato (`cal-day-outmonth`): questi giorni vengono visualizzati per rendere il formato del calendario più estetico, ma non sono selezionabili.
- Giorni appartenenti al mese selezionato (`cal-day-inmonth`): questi sono i giorni selezionabili. Cliccandoci sopra, la maschera del giorno selezionato viene caricata, rendendo possibile assegnare un voto.
- Il giorno attuale (`cal-day-today`): nella versione originale di Bootstrap Calendar, questo giorno veniva evidenziato in maniera più palese. Il calendario di RYL

Figura 3.5: Diverse visualizzazioni del calendario



(a) Visualizzazzione settimanale



(b) Visualizzazione mensile

(c) Visualizzazione annuale

[9]Cockburn, Andy, Amy Karlson, and Benjamin B. Bederson. "A review of overview+detail, zooming, and focus+context interfaces." *ACM Computing Surveys (CSUR)* 41.1 (2008): 2.

```
$(".cal-month-day.cal-day-inmonth").each(function () {
    var dayCell = $(this);
    var dayCellDate = dayCell.attr("data-cal-date");
    $.ajax({
        type: 'GET',
        url: '/RateYourLife/day/' + dayCellDate + '/vote/value/',
        datatype: 'json',
        success: function (response) {
            dayCell.addClass('vote-color-' + response.vote);
        }
    });
});
```

Listing 3.1: Query al server per colorare i giorni del calendario

offre però molti più indizi visivi all'utente, pertanto si è scelto di mettere in evidenza il giorno attuale solo colorandone il numero in modo da non sovraccaricare troppo l'interfaccia.

- Il giorno selezionato (`cal-day-selected`): il giorno che viene mostrato nel detail. Intorno a questo giorno è stata aggiunta una cornice.
- Giorno a cui è stato assegnato un voto (`vote-color-1` - `vote-color-10`): quando viene caricato il calendario viene mandata una Ajax query al server per ogni giorno del mese (v. 3.1). La risposta del server contiene il voto, in base a cui viene assegnato la classe CSS e il `background-color` appropriato.

Nel calendario sono anche visualizzati i tag assegnati (quelli positivi in verde, quelli negativi in rosso).
Essendo la simultaneità delle due visualizzazioni uno degli aspetti centrali del design overview+detail, ogni volta che viene cambiata assegnato un voto o un tag nella maschera del detail, il calendario viene attualizzato dinamicamente.

Maschera del voto

La maschera del voto è il centro dell'applicazione, implementando la maggior parte dei requisiti funzionali di 2.4. E' suddivisa in due parti principali parti (v. Fig. 3.7), quella del voto e quella dei tag. Cliccando sul rispettivo numero di stelle (1) viene assegnato il valore numerico, da 1 a 10, del voto. Il campo di testo sottostante (2) permette di aggiungere un commento facoltativo.
I tag sono divisi in due colonne, quella dei tag disponibili (3) e quella dei tag già assegnati (4). La zona blu contiene tutti i tag che l'utente può ancora assegnare al proprio giorno; la zona verde e quella rossa contengono rispettivamente i tag positivi e quelli negativi. Tutte le assegnazioni dei tag (rimuoverne uno assegnato, cambiarne la "positività", o assegnarne uno nuovo) sono possibili tramite drag 'n drop.
Tutte le iterazioni possibili nella maschera del voto non richiedono che l'utente invii esplicitamente una form, bensì sono anche esse basate su Ajax.

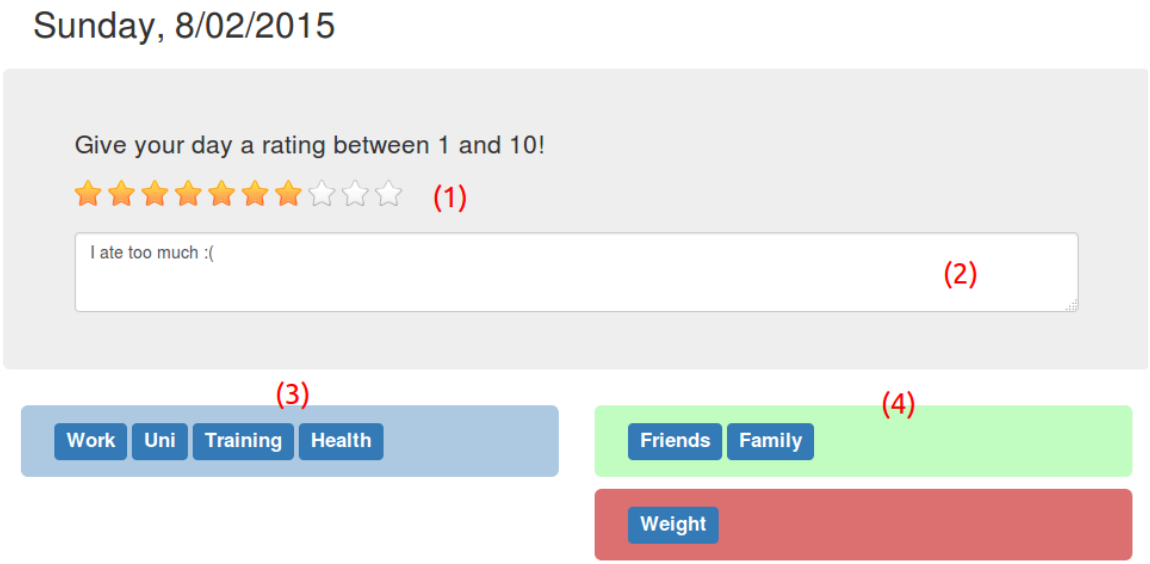


Figura 3.7: Maschera del voto

3.4 Fisheye

Detail+Overview è ottimale per risoluzioni alte. Per schermi più piccoli però, mantenere due visualizzazioni parallele diventa proibitivo. Per la versione mobile di RYL è stato perciò scelto un approccio diverso, dalla categoria *focus+context*: il fisheye. Nel focus+context sono presenti il contesto, che offre una vista di insieme, e il focus, il quale rappresenta l'informazione in primo piano. Già da questa descrizione è evidente il nesso con il calendario e il giorno selezionato.

Figura 3.8: Esempio di fisheye

Una delle tecniche principali del focus+context è la distorsione degli oggetti del contesto, in modo da poter evidenziare il focus (a differenza di un "semplice" zoom dove il contesto non è più visibile). Nel *fisheye* questa distorsione avviene in forma di uno "zoom selettivo", come mostrato da Fig. 3.8. L'implementazione da cui si è tratta ispirazione è quella del *DataLens*[10], un software sviluppato dalla Microsoft, che applica il concetto di fisheye a un calendario per computer palmari.

[10]Bederson, Benjamin B., et al. "DateLens: A fisheye calendar interface for PDAs." *ACM Transactions on Computer-Human Interaction (TOCHI)* 11.1 (2004): 90-119.

3.4.1 Implementazione

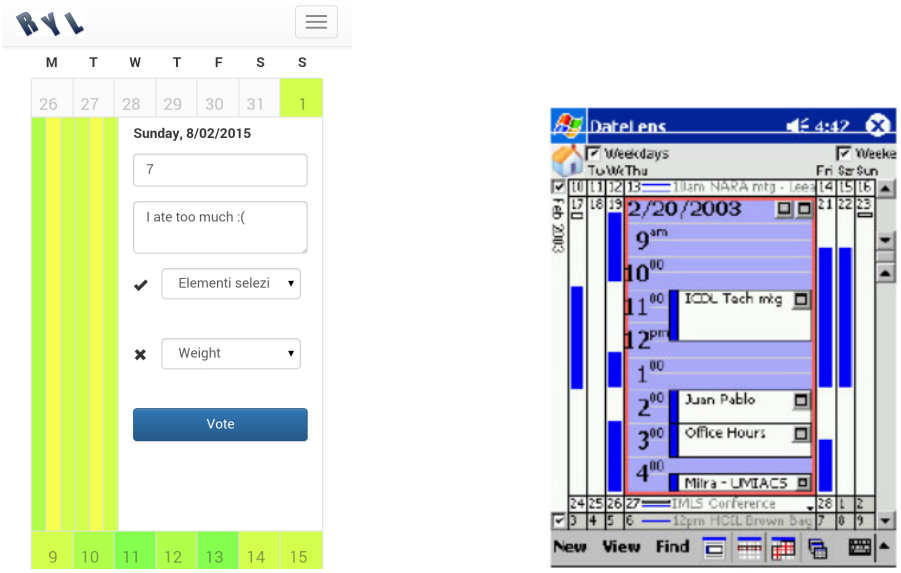
L'implementazione di fisheye è un misto di varie tecniche. Al di sotto dei 640 pixel, la maschera del voto viene nascosta tramite una media query e un flag viene settato con jQuery, mettendo il calendario in modalità mobile. In questa modalità, selezionare la casella di un giorno sul calendario non comporta l'aggiornamento del detail, bensì lo spostamento in primo piano del giorno selezionato. Questo avviene in due passi:

1. Una serie di classi CSS vengono aggiunte tramite jQuery agli elementi del calendario. Le caselle al di sopra e al di sotto del giorno selezionato vengono rimpicciolite sia in grandezza che in larghezza, mentre le caselle della stessa settimana vengono solo ristrette in larghezza; l'elemento selezionato riempe lo spazio che si è formato. Questo rappresenta l'effetto di distorsione tipico del fisheye.

2. tramite Ajax viene caricata una versione mobile del detail, che permette all'utente di assegnare voti, commenti e tag. A differenza del detail della versione desktop però, la versione mobile richiede l'invio di una form. Questa decisione e stata fatta su base di valutazioni dell'usabilità che indicavano una mancanza di feedback durante l'uso della versione mobile.

La fig. 3.9 mostra il risultato finale, paragonato a DataLens.

Figura 3.9: Implementazioni di fisheye



(a) Fisheye in RYL

(b) Fisheye in DataLens

3.5 Gestione dei tag

La gestione dei tag è una parte non tanto interessante ma necessaria dell'applicazione. Viene principalmente realizzata tramite view generiche[11], un modulo di Django che permette di creare interfacce CRUD con poco codice. Tutte le funzionalità (create, read, update, delete) sono implementate con pressapoco 100 linee di codice.

[11] Documentazione: `https://docs.djangoproject.com/en/1.7/topics/class-based-views/intro/`

4 Valutazione dell'usabilità

Questo capitolo si concentra sulla valutazione dell'applicazione dal punto di vista della usabilità. Tra i vari tipi di metodi di valutazione viene scelto quello più appropriato al prototipo. Alla fine della valutazione vengono elencati i risultati.

4.1 Metodo di valutazione

Per la valutazione dell'usabilità esistono diversi approcci, da quelli basati sugli utenti a quelli che sfruttano modelli dell'applicazione in questione. A causa della mancanza di utenti disponibili per il test e della scarsità dei mezzi idonei ad un osservazione empiricamente valida della loro interazione, si è optato per la valutazione basata sull'ispezione. Oltre ad essere più economica, questa offre anche il vantaggio di essere più adatta a prototipi e più facilmente eseguibile da una persona sola.
Il metodo di ispezione scelto sono le euristiche di Nielsen. Le euristiche offrono un insieme di regole ben definito ricavato da studi di usabilità e quindi basato sull'esperienza; di conseguenza offrono una base empirica e più oggettiva di una valutazione non schematizzata.

4.2 Risultati

Le euristiche di Nielsen offrono dieci regole da rispettare nei confronti dell'usabilità di un'applicazione.

1. Visibilità dello stato del sistema:
   - ✓ La semplicità inerente del design rende facile riconoscere cosa sta facendo il sistema.
   - ✓ Ogni pagina è fornita di titoli che ne rispecchiano la funzione.
   - ✓ Tutti gli elementi necessari per l'interazione saltano subito all'occhio.
   - ✗ La mancanza di feedback diretto in alcune query Ajax possono confondere l'utente sull'esito delle proprie interazioni.
2. Il linguaggio della applicazione riflette quello reale:
   - ✓ Tutti i bottoni o link hanno delle etichette appropriate.
   - ✓ Vengono usate le icone di Bootstrap (Glyphicons[1]), che coprono tutti gli scenari più comuni.

[1]Documentazione: `http://getbootstrap.com/components/#glyphicons`

3. Pieno controllo dell'utente:
   - ✓ Non ci sono operazioni di lunga durata che sia necessario poter interrompere.
   - ✓ Non ci sono animazioni.
   - ✓ La barra di navigazione in cima alla pagina permette all'utente di mantenere il controllo. La pagina attuale è sempre evidenziata (grazie alla classe CSS `active`).
   - ✗ L'uso di Ajax può non rendere chiaro all'utente quando lui sta comunicando con il server.
4. Coerenza:
   - ✓ L'uso delle classi CSS di Bootstrap permette di mantenere coerente l'aspetto su tutte le pagine dell'applicazione riguardo forme, dimensioni e colori.
   - ✓ Uso coerente dei colori (per esempio rosso per operazioni di cancellazione).
   - ✓ Tutte le form sono raggruppate in maniera coerente e mantengono lo stesso layout.
5. Prevenire errori:
   - ✓ Tutte le form utilizzano campi di input di HTML5 che facilitano la validazione (per esempio la lunghezza massima di un commento viene anche imposta nella `textarea` corrispondente).
   - ✓ Operazioni di cancellazione (per esempio dei tag) richiedono una conferma.
   - ✗ La grafica del calendario che rappresenta i tag assegnati ad un giorno (pallino verde o rosso) contiene un link che porta al tag in questione. Questo può venire erroneamente cliccato durante la selezione della giornata.
6. Non richiedere sforzi di memoria:
   - ✓ L'applicazione è stata sviluppata seguendo un principio di intuitività che minimizza la necessità di memorizzare funzioni.
   - ✓ Non sono presenti testi lunghi.
7. Flessibilità ed efficienza:
   - ✓ La navigazione dei giorni nel calendario rende efficiente la manipolazione dei giorni.
   - ✗ La natura dinamica del sito rende difficile l'uso di preferiti.
   - ✗ Non sono presenti adattamenti a utenti di diversi livelli di capacità.
   - ✗ Non sono implementate impostazioni modificabili.
8. Design estetico e minimalista:
   - ✓ Uso della classe CSS `hero unit` per evidenziare la parte principale del voto.

- ✓ Il layout rete usato risulta in un sito ben raggruppato.
- ✓ Assenza di testi troppo lunghi.
- ✓ Uso di Bootstrap rende l'estetica del sito moderna e piacevolmente familiare grazie alla diffusione dello stile di Bootstrap.
- ✗ La gestione dei tag è alquanto maccheronica.

9. Aiutare gli utenti nella gestione di situazioni erronee:
   - ✓ Implementazione di una pagina `error.html` a cui viene ridiretto in caso di errore e dove possono essere visualizzati messaggi inerenti all'errore occorso.
   - ✗ La natura dinamica del sito rende l'uso del pulsante "indietro" del browser poco utile.
10. Fornire aiuto e documentazione:
    - ✓ Tutti gli input sono ben descritti.
    - ✗ Mancanza di un FAQ.
    - ✗ Mancanza di un tour di presentazione quando viene acceduto il sito per la prima volta.

5 Conclusione

Nel corso di questa relazione si è cercato di affrontare le sfide delle applicazioni multi-dispositivo con l'uso di tecnologie moderne. Il risultato è Rate Your Life, un'applicazione in grado di adattarsi e regolare le possibilità di interazione in base al dispositivo in uso. Si spera che questo prototipo possa essere in parte visto come una validazione dei concetti di progettazione di interfacce e valutazione dell'usabilità applicati.